Protecting Temporal Fingerprints with Synchronized Chaotic Circuits

Fengyi Tang[1,2] and Betty H.C. Cheng[1,3]
[1]Department of Computer Science and Engineering, Michigan State University
[2]College of Osteopathic Medicine, Michigan State University
[3]Software Engineering and Network Systems Laboratory
{tangfeng, chengb}@msu.edu

Abstract

In recent years, connected autonomous vehicles (CAVs) feature an increasing number of Ethernet-enabled electronic control units (ECUs), thereby creating more threat vectors that provide access to the Controller Area Network (CAN) Bus. Currently, mitigation techniques to protect the CAN bus from compromised ECU units in vehicle ad hoc networks (VANET) often utilize classical cryptographic techniques. However, ECUs often have temporal signatures that leak internal state information to eavesdropping attackers who can leverage temporal properties for longitudinal attacks. Unfortunately, these types of attacks are difficult to defend against using classical encryption schemes and intrusion detection systems (IDS) due to their high computational demands and ineffectiveness at protecting CAVs throughout the duration of their long lifespans. In order to address these problems, we propose a novel cryptographic framework that protects information embedded in ECU network communications by delivering an encryption system that periodically "salts" the temporal dynamics of individual ECU units with chaotic signals that are difficult to learn. We demonstrate the framework on two datasets, and our results show that the underlying temporal signatures cannot be approximated by state-of-the-art learning algorithms over finite time horizons.

1. Introduction

The design of effective encryption techniques has long been a difficult problem in cyber-security research for connected autonomous vehicles (CAVs) [1]. In recent years, CAVs feature an increasing number of Ethernet-enabled electronic control units (ECUs), creating more threat vectors that provide access to the Controller Area Network (CAN) Bus. Currently, mitigation techniques to protect the CAN bus from compromised ECU units in vehicle ad hoc networks (VANETs) often utilize classical cryptographic techniques. However, due to increasing computational power and the rising number of automated attack algorithms, mitigation by classical cryptographic techniques can be ineffective in protecting CAVs throughout the duration of their long lifespans. Manufacturers can, however, purchase expensive intrusion detection systems (IDS) as a second-line of defense. Because an IDS can be computationally expensive to embed into ECU networks, the may be difficult to scale in mass production. Additionally, ECU units often leak temporal information about their internal states (e.g., clock off-sets, odometer readings). By observing the change in internal states across time, one can approximate a set of temporal dynamics that generate these "temporal signatures." Attackers that can learn the temporal signatures of ECU units can launch longitudinal attacks that are difficult to detect by IDS. This paper introduces a novel encryption framework that protects the ECU network integrity by embedding temporal signatures of ECUs in chaotic signals that are updated aperiodically, and thus more difficult to learn.

Recent work in the automotive domain have demonstrated the capacity for automated algorithms to recover hashes designed by classical cryptographic techniques in relatively short time periods [2, 3]. Moreover, classical cryptographic techniques cannot protect against a wide range of ECU temporal signals that are prominent in a communication network. For example, low-level ECUs often leak signals with periodic signatures, thus allowing for easy spoofing and denial of service (DoS) attacks on critical units. Thus, the design of an effective encryption system must not only provide a secure authentication mechanism, but it must also protect against the propagation of compromised units during incremental attacks.

A key observation is that there exists a commonality among attacks that possess the propensity to escape classical encryption and IDS systems: the attacker learns some unprotected temporal signals governing critical ECU units in a network and then incrementally

URI: https://hdl.handle.net/10125/64519
978-0-9981331-3-3
(CC BY-NC-ND 4.0)

modifies the underlying temporal off-sets in a way that remains undetected by the system's anomaly detection mechanism. The key assumptions enabling such attacks can be summarized as follows: (1) the attacker must be able to *approximate* the temporal patterns of a subset of ECU units, which presumably follow a set of fixed periodicities, and (2) the attacker can increase the noise according to a smooth function such that the IDS system cannot detect the change in baseline over a long period of time. In order to address these challenges, we introduce a dynamical encryption system that adaptively "salts" the temporal dynamics of individual ECU units with aperiodic signals that are difficult to learn.

We present a technique to mask the original ECU signatures within an ECU network with a set of chaotic dynamics, synchronized across the VANET. We also equip this dynamical encryption mechanism with an aperiodic update algorithm that updates the salting mechanism in a way that makes it difficult to approximate using state-of-the-art learning algorithms. We show that aperiodicity is needed to deal with the long lifespans of CAVs, which makes classical encryption mechanisms susceptible to brute force algorithms when given enough time for computation.

We validate our framework on two datasets: a synthetic dataset involving temporal signals following chaos dynamics [4, 5], and a benchmark dataset from the University of California Irvine (UCI) machine learning repository [6]. Our results demonstrate that the proposed encryption system can dynamically mask temporal signals in a way that cannot be approximated by state-of-the-art learning algorithms over finite time horizons. The remainder of this paper is organized as follows. We summarize related work in Section 2. Section 3 characterizes the so-called "nudging attacker" that embodies the key limitations of modern security approaches for ECU protection. We formalize the problem of encrypting ECU communication channels in Section 4. The proposed method is described in detail in Section 5. Experimental results are presented in Section 6. Key conclusions and future directions are discussed in Section 7.

2. Related Work

Conventional encryption techniques such as elliptical curve cryptography (ECC), public-key cryptography (PKI), and symmetric key cryptography (e.g., AES) are limited in their applications in VANETs due to memory constraints, high computational cost, and limited adaptability to ad hoc changes in VANET topologies [7, 8]. Recent advances in lightweight PKI (e.g., [9]) has alleviated portions of the computational overhead, allowing for conventional approaches to be applied in the ECU and VANET settings. However, Verdult et al. [10] showed that brute force algorithms, given state-of-the-art computational power, can compromise VANETs that are encrypted by Megamos Crypto algorithm (used by many manufacturers) in seconds. Thus, encryption keys need to be frequently updated over the lifetime of CAVs, a costly process using classical methods. Furthermore, the state information of ECUs are often high-dimensional and span across numerous time-steps; the temporal nature of the data presents a unique challenge for classical encryption schemes.

Recently, Cho et al. demonstrated that IDS based on clockwork off-sets can detect intrusions based on deviations from standard ECU fingerprints [11]. However, the proposed IDS is based on regressive least squares (RLS) and detects anomalies that deviate from baseline in accordance to linear transition dynamics. An attacker can thus incrementally increase the attack intensities to perturb the clockwork off-sets in a way that dominates the mean ECU temporal signatures without detection by the RLS system. For example, Cho et al. alluded to the possibility of a "weak attacker" whose main goal is to eavesdrop on the signals produced by various ECUs in a network in order to obtain ECU temporal fingerprints. Thus, a potential solution should offer an encryption system that protects the temporal signatures along the time dimension in order to prevent long-term synchronization of its intrinsic periodicities with attackers.

3. The Nudging Attacker

In this section, we formulate the problem setting of ECU communications into a dynamical systems framework and introduce the characteristics of the incremental attacker – we call it the "nudging attacker". We present the nudging mechanism as a circuit synchronization problem where the attacker can observe and learn the true dynamics of the driver ECU circuit and incrementally change its update rules through low-amplitude messages.

3.1. Notation

Here, we overview key notations used to describe the problem setting and our proposed technique. We denote $x(t) \in \mathbb{R}^n$ to be the state vector representing the internal state of an ECU at time $t \in [0, \infty)$. We denote $\dot{x} = F(x)$ to be the update dynamics of the ECU and denote $\hat{x}$ as the *perceived* dynamics of the receiver, which in this case is the CAN Bus. Similarly, we denote $\hat{F}$ as the *approximated* dynamics from receiver's point

of view. Note that x and $\hat{x}$ may differ due to inherent noise in the communication channels. Similarly, F and $\hat{F}$ correspond to the same function when the receiver is the CAN Bus. However, if the receiver is an *attacker*, then $\hat{F}$ may not be known perfectly as in the CAN Bus case, and large deviations where $\hat{F} \neq F$ leads to de-synchronization.

3.2. ECU Communications as Synchronizing Circuits

We formalize the problem setting by considering ECU communications in a VANET as a synchronization problem between circuits that operate by state dynamics that can be described by ordinary differential equations. Consider, for example, the communications between an arbitrary ECU unit and the CAN Bus:

$$\dot{x}_1 = F_1(x(t)) \quad \dots \quad \dot{x}_n = F_n(x(t)) \quad \text{(Driver Circuit)}$$

Here, the "driver" component is the ECU unit that sends characteristic messages $x_1(t) \in \mathbb{R}$ to the "receiver" circuit that is the CAN Bus. For example, $x_1(1), \dots, x_1(T)$ can be a sequence of continuous signals that carry critical information regarding the ECU functionalities to be integrated into the CAN Bus for prioritization, broadcasting, etc.

$$\dot{\hat{x}} = \hat{F}(y, \hat{x}_2(t), ..., \hat{x}_n(t))$$
$$y(t) = x_1(t) \qquad \text{(Receiver Circuit)}$$

The receiver circuit represents the CAN Bus, which, in reality, only observes the $x_1(t)$ component of the ECU state. The CAN Bus modifies its internal dynamics $\hat{F}$ to approximate the actual dynamics of the ECU. Finally, we define *synchronization* between circuits to be a condition where

$$\lim_{t \to \infty} ||x(t) - \hat{x}(t)|| = 0. \tag{1}$$

Under this formulation, we can observe several important characteristics. For example, the CAN Bus only observes a subset of the state variables (i.e., $x_1(t)$), but the entire dynamics of the ECU circuit may depend on several state variables – that is, the system dynamics $F(x)$ depends on n-distinct state variables $x \in \mathbb{R}^n$. In reality, even when an encryption system is placed on the CAN Bus, they typically protect the state variable in direct communication between the CAN Bus and the ECU unit of interest [12]. However, other components of x (e.g., clock off-sets, computational usage, energy output patterns) can often leak important information regarding the state dynamics of the ECU system [11].

3.3. Attacker Dynamics

Suppose an attacker has access to several state variables that follow temporal patterns (e.g., fingerprint information of $x_2, \dots, x_n$) [11], an attacker can mimic the baseline behavior of the ECU unit – this is often called a "weak attacker" [11]. Under this setting, the attacker, when given enough time, can approximate x_1 from the approximate state dynamics $\hat{F}$ by state-of-the-art learning algorithms (e.g., extended Kalman filters (EKFs), RLS algorithms, recurrent neural networks) [13, 14]. At this point, we can define the modified system with a learned attacker as follows:

$$\dot{x} = F(x(t))$$
$$s(t) = x_1(t) + m(t) \qquad \text{(Driver Circuit)}$$

Here, the attacker's dynamics closely mimic those of the original ECU, with the exception of the $s(t) = x_1(t) + m(t)$ component. As a result, the CAN Bus now receives a competing signal:

$$\dot{\hat{x}} = \hat{F}(s(t), \hat{x}_2(t), ..., \hat{x}_n(t))$$
$$y(t) = s(t) \qquad \text{(Receiver Circuit)}$$

The receiver circuit cannot distinguish between $s(t)$ and $x(t)$, given that $||m(t)|| < \varepsilon$ for some threshold value ε. Given the same set of initial conditions, Pecora et al. [15] and Cuomo et al. [16] showed in previous studies that with the choice of Lyapunov function $V(e) = \frac{1}{2}(p_1 e_1 + p_2 e_2 + \cdots + p_n e_n)$ and properly chosen parameters $p_1, \dots, p_n$, the *error dynamics* $e_1 = x_1(t) - s(t), \dots, e_n = x_n(t) - \hat{x}_n(t)$ become globally exponentially stable and decays toward the origin in finite time. In other words, the attacker can synchronize with the CAN Bus while delivering additional messages $m(t)$ without desynchronizing the overall communication between the ECU unit and the CAN Bus. In Pecora et al. [15], the authors demonstrates that $m(t)$ can actually be used to send messages to be decoded by the Receiver Circuit.

In our case, depending on what type of information is chosen in $m(t)$, the attacker can deliver various downstream effects on the ECU and CAN Bus communications. For example, at some time $t > 0$, the attacker may choose to inject a large noise $||m(t)|| >> \varepsilon$ that desynchronizes the systems, resulting in DoS. However, a sudden increase in $m(t)$ (non-smooth or discontinuous) can easily be detected by an IDS that detects anomalous signals outside of baseline behaviors [11]. Thus, we define a *nudging attacker* as one that *progressively* increments $||m(t)||$ slowly, in a way that remains within the baseline behavior of $x(t)$ as to not be detected by the IDS.

4. Protection of ECU Communications

To address the unique challenges posed by the nudging attacker, we propose an encryption scheme based on choosing the right masking signals at each time-step to prevent attackers from learning the underlying fingerprints of ECU signals. We embed the original ECU signal in a masking signal that follows non-stationary transition rules across time. The masking signal is unpredictable by approximators over finite time-horizons, and the underlying signal cannot be recovered without knowing the update and transition rules, that is, the private keys – beforehand.

We present the main characteristics of encryption blocks that are used to protect the system messages. Then we present the key concepts that underlie the security of the encryption blocks in the context of predictability of temporal signals. We also introduce an update mechanism for the encryption blocks so that attackers have only finite time to decipher their parameters, despite the long lifespan of CAVs.

4.1. Masking Signals with Encryption Blocks

In general, an *encryption block* can be described as a dynamical system

$$x(t+1) = G(x(t), z) \tag{2}$$

where $x(t)$ is the *cryptogram* at time-step t, $x(0)$ is the *plaintext*, $G(.)$ serves as the hash function, and z is the *private key* that allows for $x(0) = G^{-t}(x(t), z)$ to be recovered. Here, $G^{-t} = G^{-1} \dots G^{-1}(.)$ is the repeated application of the inverse of G, t-times. The key idea is that without z, the inverse hash G^{-1} is difficult to be approximated by a polynomial-time algorithm [17]. Previously, Kocarev et al. [17, 18] investigated the characteristics of classical hash functions (i.e., based on modular arithmetic) and chaotic systems (i.e., chaos dynamics in continuous space) to be used as $G(.)$.

Using chaos dynamics in encryption blocks offers several notable advantages. ECU signals that are leaked to attackers are continuous (i.e., $x(t) \in \mathbb{R}^n$) and spans across multiple timesteps. Although classical cryptographic hashing functions can be applied to scalar inputs, such an approach has to consider the collision rate between similar states, which is difficult to assess in high-dimensional state space.

To apply a chaos encryption block to our problem, we treat F as the original ECU dynamics to be communicated to the CAN Bus. A chaos encryption block G masks the original dynamics as follows:

$$\dot{x} = F(x(t)) + G(s(t), z) \qquad \text{(Encrypted Driver)}$$

where F is the original dynamics of the driver signal and G is the encryption block dynamics that has mixing [19] and diffusion properties [18]. The idea behind the mixture property is that $G(.)$ must be *volume-preserving* with respect to the search space of possible states for synchronization, and the diffusion property allows for neighboring states to be "spread-out" across time by the *chaos dynamics* of G [17]. Pecora et al. [15] showed a simple example using Lorenz circuits for $G(.)$. The CAN Bus can thus decode the true ECU message $\hat{x}(t) = F(\hat{x}(t))$ upon receiving $x(t)$:

$$\begin{aligned} \dot{\hat{x}} &= F(\hat{x}(t)) \\ \hat{x}(t) &= x(t) - G(x(t), z) \end{aligned} \qquad \text{(Decoding Receiver)}$$

However, a bottleneck for chaos encryption remains: *under what conditions can the chaos dynamics be unpredictable by probabilistic polynomial-time algorithms* [17]?

4.2. Predictability of Chaotic Signals

To evaluate the security of an encryption block, the predictability of $G(.)$ must be examined. Classical encryption algorithms such as SHA-256 protect $G(.)$ by minimizing the collision rate of the hash function outputs, which serves to limit the computability of the pre-image [20]. For continuous dynamics, however, an alternative to cryptographic security must be taken. Chaos dynamics exhibit favorable properties to this end: neighboring points in the state space diverge in their trajectories (i.e., diffusion property), and exact trajectory recovery cannot be done with deterministic algorithms [17]. However, it is unclear whether polynomial-time approximation algorithms can recover chaos dynamics when given finite horizons for evaluation. For example, Ishii et al. demonstrated that online approximation algorithms and recurrent neural networks can approximate Lorenz and Kuramoto-Sivashinksy dynamics based on observed data over finite time horizons [21]. Thus, we examine the "predictability" of an encryption block from the point of *learnability*: How easy is it to learn $G(.)$ and $G^{-1}(.)$ given full observation of past trajectories, with continuous $G(.)$ and $x(t) \in \mathbb{R}^n$?

From the information theory perspective, *learnability* relates to Yao's definition of *computationally unpredictable* [22], which identifies a random process as "unlearnable" if a probabilistic approximator cannot do better than random guess to predict the initial sequences of inputs into the random process. Alternatively, modern learning theory identifies "learnability" of a random process as having bounded *generalization error* that monotonically decreases with

increasing training samples [23]. In our evaluation, we consider both definitions: given fully observed historical information, a state-of-the-art approximator should *generalize poorly* when predicting the future states governed by the encryption dynamics.

4.3. Salting

Computational resources must be taken into consideration for *learnability* [17]. For example, an attacker with infinite amount of time can crack a SHA-256 algorithm using L bits by exploring the 2^L search space using a brute force mechanism. To deal with brute force attacks with long time horizons for calculation, classical methods involve adding a "salt" to the original hash to periodically update the underlying encryption mechanism. "Salting" is a process whereby the encryption block is turned into an iterative hashing procedure [24]:

$$\begin{aligned} x(t+1) &= G(s(t), z) \\ s(t) &= H(x(t), \zeta). \end{aligned}$$

Here, $H(.)$ serves as a second hash function that "salts" the pre-image (according to some secret key ζ) before input into the encryption block. We use a similar strategy to ensure that the statistical properties of the encryption blocks evolve over time.

5. Methodology

In this section, we present the main components of our methodology. We start with the basic encryption and decoding blocks. Then we formulate the update mechanisms for salting and encryption dynamics. Finally, we introduce a synchronization scheme that distributes the private keys of specific encryption blocks throughout an ECU network.

5.1. Encryption and Decoding Blocks

The encryption block for each ECU unit follows:

$$\begin{aligned} \dot{x} &= F(x(t)) + G(x(t), z(t)) && \text{(Encrypted Dynamics)} \\ \dot{m} &= H(m(t), z(t)) && \text{(Salting Update)} \\ \dot{z} &= S(x(t), m(t)) && \text{(Dynamics Update)} \\ s(t) &= x(t) + m(t) && \text{(State Salting)} \end{aligned}$$

where $G(.)$ is the hash function that operates on chaos dynamics, $H(.)$ is the salt update function, and $S(.)$ is the dynamics update function and $s(t)$ represents the state information, after salting, that is communicated to the CAN Bus. Here, we set several constraints on $G(.)$:

- $||G(x(t), z(t)|| >> ||F(x(t))||$ is required to successfully "mask" the original dynamics of the system F.
- $z(t)$ parameterizes G, and it is updated across time.

The key observation about the encryption dynamics is that because F is periodic, it is easily learned by an attacker based on observation. However, when the magnitude of the encryption dynamics greatly exceeds those of the original periodic signals, F is effectively "masked" as noise under G.

From the receiver point of view, the decoding block (i.e., CAN Bus, potential attackers) can be formulated as follows:

$$\begin{aligned} \dot{\hat{x}} &= \hat{F}(\hat{x}(t)), & \hat{x}(t) &= s(t) - \hat{G}(s(t), \hat{z}(t)) \\ \dot{\hat{m}} &= \hat{H}(\hat{m}(t), \hat{z}(t)), & \dot{\hat{z}} &= \hat{S}(s(t), \hat{m}(t)) \end{aligned}$$

where $\hat{F}, \hat{G}, \hat{H}, \hat{S}$ are to be *estimated* by the attacker, but are *fully observed* by the actual CAN Bus. Since $z(0)$ (private key) and $m(0)$ (salt) are fully known by the CAN Bus, the true receiver of the ECU information can recover the actual trajectory of the ECU dynamics using the update equations given at manufacturer settings. The attackers must not only recover the hash functions G, H, S and F, they must also estimate correctly the $z(t)$ and $m(t)$ keys at each time step, or they will desynchronize from the ECU state dynamics over time.

Since z parameterizes G, updating $z(t)$ across time changes the dynamics of G. The *learnability* of G depends on parameters in the estimation problem:

- The *sample complexity* of the approximation algorithm used to learn G, which is defined as the *number of samples required to achieve a particular generalization error* [23].
- The number of observations available to learn G.

The key contribution of S is that one can control the effective sample size of historical information available to the attacker. For example, if S updates $z(t)$ every T timesteps, then the attacker only has $x(1) \dots x(T)$ valid historical information to learn G. Beyond timestep T, the dynamics for G is reset, rendering the historical information of state trajectories $x(1), \dots, x(T)$ invalid to predict $x(T+1), \dots, x(2T)$.

As a case in point, suppose the attacker utilizes a highly complex approximator such as a deep long-term short-term memory (LSTM) network [25]. Such models require a large sample size in order to achieve reasonable generalization error due to high model complexity [23].

Furthermore, the latent states of LSTMs are updated according to stationary weights with learned "forget gates" from data. However, if S is selected in such a way that $x(1), \ldots, x(T)$ does not provide adequate training samples for the attacker model, then it will lead to *overfitting* [23] and poor generalization to predictions over future time horizons. Thus, the purpose of S is to decouple the state dynamics from historical trajectories, thus *preventing the learnability of the encryption block over long time horizons*.

5.2. Selection of Salting and Update Dynamics

The main issue concerning the selection of G, S and H is to ensure that while G itself may be approximated by an attacker, H and S can update G in a way that makes it *computationally unpredictable*. When S is used to update z, the dynamics of the system change according to some time horizon related to the periodicity of S. The attacker trying to model the time-series data must adapt its approximations based on the change in $z(t)$. This setting is similar to *online learning* of time-series [26], which describes the learning problem where the training and testing trajectories come from different (but related) distributions. A successful online-learning algorithm must adapt the algorithm to handle shifting distributions in the data stream.

Thus, the problem of selecting S and H becomes one of updating the online-learning tasks in a way that induces *catastrophic forgetting*, which is defined as abrupt decrease in performance on learned tasks due to large drifts in the data stream distributions [27–29]. Catastrophic forgetting typically results from *negative transfer* between previously learned trajectories and the new trajectory [30]. For this reason, the design of S and H considers a set of operators (e.g., modulus) that can induce discontinuity in the dynamics of $z(t)$ while maintaining the chaos dynamics (i.e., diffusion and mixing properties) of G. Although G can be approximated by various estimation algorithms, we explore several candidates for the selection of S and H to obfuscate the dynamics of G across time.

6. Experiments

We validate our proposed framework on synthetic data as well as benchmark data obtained from University of California Irvine Machine Learning (UCL-ML) Dataset [6]. In synthetic experiments, we investigate the feasibility of masking simple signal dynamics with salted chaotic dynamics in the batch-learning and online-learning settings. In the benchmark experiments, we apply the full encryption blocks and investigate the application of the Driver-Receiver system. On these datasets, we create a Driver Circuit that sends the encrypted data, a Receiver Circuit that decodes the encrypted data, given the initial conditions (plaintexts), parameterizations of the encryption blocks (private keys), as well as the update and salting dynamics.

6.1. Signal Masking with Chaos Dynamics

We first demonstrate the encryption mechanism on synthetic time-series data from the UCI ML dataset [6], specifically the *Pseudo Periodic Synthetic Time Series Data Set*. This dataset features $100,000$ samples of multivariate time-series data, generated from the stochastic periodic function:

$$f(x) = \sum_{i=3}^{7} \frac{1}{2} \sin\{2\pi(2^{2+i} + \text{rand}(2^i))\bar{t}\}. \quad (0 \leq \bar{t} \leq 1) \tag{3}$$

We can see an example of the time-series signal in Figure 1. In total, there are 10 channels with fixed-time intervals (i.e., $dt = 0.1$). The number of time-steps for each sample is fixed at $t = 10$. Thus, we formulate the learning problem as time-series forecast whereby the first 9 time-steps for each sample in the training set is used as the features (X), and the last time-step of each sample is used as the target (y).

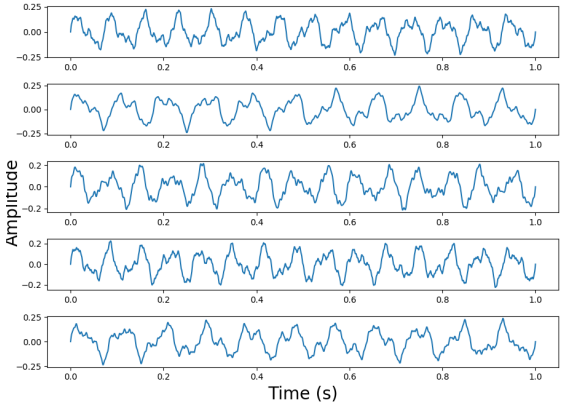


Figure 1: Pseudo Periodic Signals for UCI Dataset [6]

Under this setting, there are two modes of evaluation: batch-learning and online-learning [23]. In batch-learning, we apply the classic $70 - 30$ training- and testing-split over 5 randomized shuffle splits. During testing, the models are not adjusted. In the online-setting, we treat dataset as a *data stream*, whereby the model sees one test sample at a time and *is allowed to adjust its parameters* to better predict incoming samples from the data stream.

Table 1 illustrates the performance of various

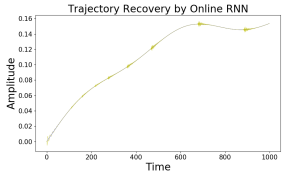
state-of-the-art approximators for time-series prediction, including the recurrent neural network (RNN) [31] and the long short-term memory neural network (LSTM) [25]. RNNs and its variants underlie much of the state-of-the-art temporal models in the machine learning community. When comparing the batch-learning results, we see from Table 1 that RNN and LSTM produced comparable performance on test sets, with LSTM slightly overfitting the univariate data. As expected, the online-learning setting allowed both models to track the signal perfectly, converging to zero-error dynamics in finite time. This result is likely due to the fact that the dynamics of our temporal signal is *stationary* throughout the data stream – i.e., there is no distribution mis-match between training and testing distributions across time.

Table 1: Predicting periodic signals with state-of-the-art approximators.

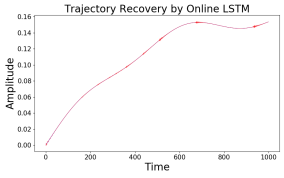
Model	Mode	MSE	RMSE
RNN	Batch	$4.9e-3 \pm 8e-8$	$0.068 \pm 3e-4$
RNN	Online	$6.6e-7 \pm 1e-11$	$2.8e-4 \pm 5e-7$
LSTM	Batch	$6.8e-4 \pm 2e-8$	$0.026 \pm 8e-6$
LSTM	**Online**	$\mathbf{8.3e-8 \pm 1e-13}$	$\mathbf{1e-4 \pm 7e-8}$

Abbreviations: *LSTM* = Long Short-Term Memory recurrent neural network, *RNN* = Recurrent Neural Network.

Figures 2a and 2b further illustrate differences between online LSTM and online RNN. Both converge to the true trajectory at roughly the same rate, but RNN acquires more chattering during trajectory recovery.



(a) Trajectory Recovery by RNN Online-Learning



(b) Trajectory Recovery by LSTM Online-Learning

Next, we illustrate the effect of encrypting the original state dynamics with chaos dynamics provided by the *Lorenz96* system [4], which is defined as follows:

$$\frac{\partial x_i}{\partial t} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F. \quad \text{(Lorenz96)}$$

The key observation about the Lorenz96 system is that the chaos dynamics (G) can scale to arbitrary dimensions. Here, we applied $N = 36$ dimensional state with a force factor $F = 8$, which has been shown to induce chaos behavior across the various dimensions [4]. We plot the first three dimensions of the Lorenz96

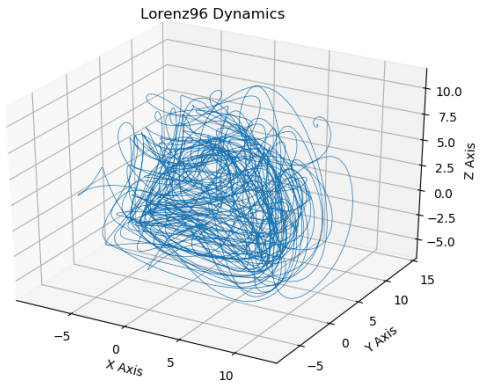


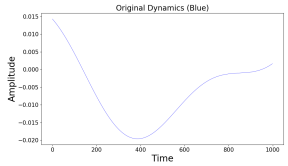
Figure 3: Illustration of the First 3 States of Lorenz96 Dynamics [4], parameterized by $N = 36, F = 8$.

dynamics in Figure 3. Applying the first component of the signal, we now have a different set of dynamics, as shown in Figures 4b-4d. In Figure 4a, we have the original signal, following $\dot{x} = F(x)$, across 1000 timesteps. The original Lorenz96 dynamics (G) is shown in Figure 4b. When we apply the encryption block $\dot{x} = F(x(t)) + G(x(t), z)$ (See Fig. 4b), we notice that the resulting dynamics differ from both the Lorenz96 and original signal dynamics (Fig. 4c). In fact, there is a phase-shift from the Lorenz96 dynamics to the new encrypted dynamics, which alters the predictability of the original chaos system (See Fig. 4d).

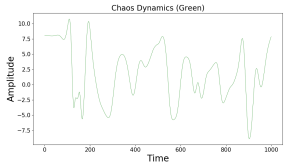
Furthermore, we note the difference in *amplitude* between the original signal (blue) and the chaos dynamics (red and yellow) in Fig. 4d. Attackers must not only infer the original signal based on the aperiodic phase-shift of the modified chaos dynamics, but they must also recover a signal that is much lower in amplitude compared to the dominating dynamics. The attacker, for example, sees only the encrypted signal (Figure 4c) and tries to recover the original signal F. However, the composition of the $F+G$ embedded signal results in a different set of dynamics that follows its own aperiodic update rules rather than a simple addition of the original signals at each time-step. By maintaining the large magnitude of chaotic signals, the new $F + G$ dynamics make recovering F difficult for the attacker.

6.2. Predictability of Salted Chaos Dynamics

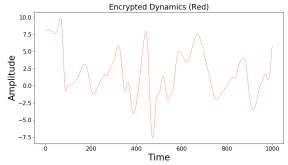
We further investigate this decoupling effect by investigating the predictability of chaos dynamics under composition with other salting functions. First, we examine the case where the chaos function G is assumed to be a simple 3-body chaos system following the Lorenz equations, described in Pecora et al. [15]



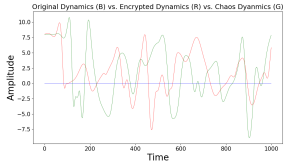
(a) Original Signal



(b) Chaos Dynamics



(c) Encrypted Signal



(d) Comparison of Dynamics

(named Lorenz3). Previous studies have illustrated its predictability using methods such as the recurrent neural network [32]. We verify that this the case for both the Lorenz3 and the Lorenz96 dynamics.

Table 2: Synthetic experiments involving the prediction of Lorenz dynamics using various state-of-the-art temporal approximators.

Model	Dynamics	Time	MSE	RMSE
RNN	Lorenz3	5	$0.017 \pm 4e-4$	$0.130 \pm 6e-4$
RNN	Lorenz3	10	$0.013 \pm 2e-5$	$0.114 \pm 3e-4$
RNN	Lorenz96	5	$0.001 \pm 1e-7$	$0.035 \pm 2e-5$
RNN	Lorenz96	10	$7e-3 \pm 2e-8$	$0.026 \pm 9e-6$
LSTM	Lorenz3	5	$0.014 \pm 3e-5$	$0.112 \pm 4e-4$
LSTM	Lorenz3	10	$0.0087 \pm 3e-5$	$0.106 \pm 8e-5$
LSTM	Lorenz96	5	$0.001 \pm 2e-8$	$0.033 \pm 4e-6$
LSTM	**Lorenz96**	**10**	$\mathbf{5e-3 \pm 2e-9}$	$\mathbf{0.023 \pm 1e-6}$

Abbreviations: *LSTM* = Long Short-Term Memory recurrent neural network, *Lorenz3* = classical 3-state Lorenz circuit, *Lorenz96* = 96-dim. chaos system based on Lorenz 1996 update equations.

To investigate the effectiveness of salting, we include static salting function to produce $m(t)$ and evaluate the system in both the batch learning and online learning settings. Specifically, we alter the dynamics of the Lorenz96 system as follows:

$$\frac{\partial x_i}{\partial t} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + z(t)$$

$$\frac{\partial z}{\partial t} = (z_i + \lfloor x(t) \rfloor)\%p + 8. \qquad \text{(Salted Lorenz)}$$

Here, we denote $\lfloor x(t) \rfloor$ as the *floor* of the Euclidean norm of $x(t)$ at timestep t. The key here is that we treat the original *force constant* of the Lorezn96 as a parameterized variable $z(t)$, and we update $z(t)$ according to a discontinuous, non-smooth hash function $\dot{z}$. The modulus term (by some interger p) allows for discontinuity in the update dynamics for $z(t)$, yet it is maintained by a force constant factor of $F = 8$ to allow chaos dynamics in G.

Since we are evaluating this method in an online setting, there is no train-test split. Instead, we set 6500 training points and allow 3491 sample data-stream during evaluation. The learning model sees one sample at a time during the evaluation phase and adapts its parameters according to the ground truth. We use RNN and LSTM models with look-back window of 10 time-steps. As illustrated in Table 3, we see that the salting dynamics is much harder to predict than the original one. While G itself can be estimated with high accuracy, G with updates on z (following a periodic $\dot{z}$ in this case) impairs the ability of online learning to recover the dynamics. We introduced

Table 3: Prediction of adaptively salted Lorenz dynamics in the Online Learning Setting.

Model	Dynamics	MSE	RMSE
RNN	Lorenz3	1.39 ± 8.67	0.851 ± 0.665
RNN	Lorenz96	0.317 ± 0.391	0.411 ± 0.149
LSTM	Lorenz3	5.03 ± 133	1.58 ± 2.52
LSTM	**Lorenz96**	$\mathbf{0.241 \pm 0.277}$	$\mathbf{0.345 \pm 121}$

non-smoothness in the update dynamics for $\partial z/\partial t$ to illustrate that gradient-based algorithms such as the RNN can experience large gradients in these situations and, as a result, the solution can diverge from previous equilibrium points.

6.3. Driver-Receiver Synchronization

To evaluate the capacity for synchronization, we apply the full encryption block on time-series data and investigate both the performance of *trajectory recovery* on the part of the Receiver, who receives a salted version of the encrypted state. As demonstrated in the previous sections, the salted dynamics follow a completely different set of transition rules compared to the original signal and the chaos signals. From the point of view of the attacker, it is difficult to recover the original signal. However, we also need to ensure that the receiver is capable of recovering the original signal F by synchronizing its updates with $F + G$ at each timestep.

We first illustrate the schematics of our experimental setup. Driver dynamics indicate the outbound communication channel that sends $s(t) = x(t) + m(t)$ using the Encrypted Driver dynamics. The Receiver dynamics indicate the inbound communication that decodes $s(t)$ into $\hat{x}(t)$ according to the Decoding Receiver dynamics. In reality, the Receiver receives a *noisy version* of $\hat{s}(t) = x(t) + m(t) + \varepsilon$, which contains a noise term $\varepsilon \sim \mathcal{N}(\mu, \sigma)$. These experiments were conducted under Gaussian noise at the level of $\mathcal{N}(0, 0.05)$. In this case, we consider the Receiver problem as one of *robust trajectory recovery*, since small

amounts of noise can de-couple synchronizing circuits.

Table 4 illustrates the performance trajectory by the Receiver circuit. Here, we evaluate performance by the mean squared error and variance between the original trajectory $x(0), x(1), \ldots, x(t)$ against $\hat{x}(0), \ldots, \hat{x}(t)$ obtained by the Receiver. We also compare the absolute difference between the encrypted signal $s(0), \ldots, s(t)$ (containing the composition of signals) against the original signals to show the difference in magnitude and periodicity under encryption. We can see that the original signal was recovered with low mean squared error and variance. Moreover, we see that both the internal states (i.e., original signal of the Drive and the decoded signal of the Receiver) differ greatly from the encrypted signal (denoted as "Driver Output"), which is seen by the public.

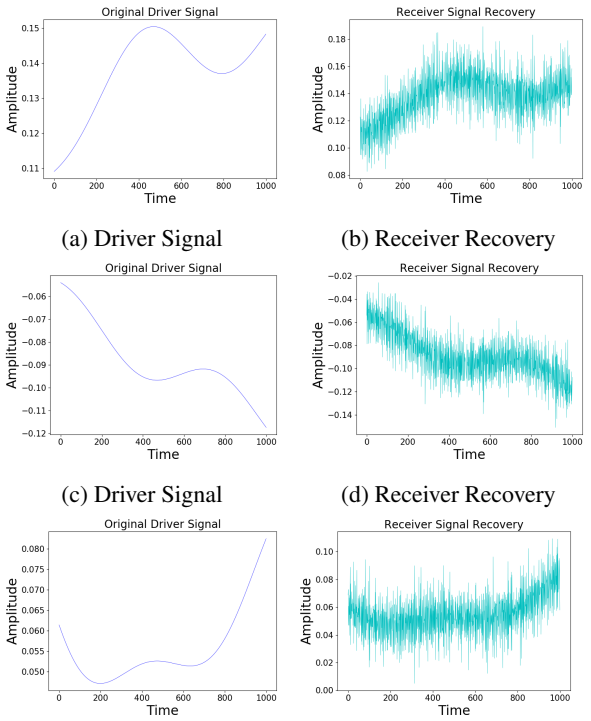
To visualize the performance of the robust trajectory recovery, we see from Figure 5a-Figure 5e three different cases. All trajectories were sampled randomly from the test set. We see that the underlying pattern is recovered in all cases. Although we note the existence of *chattering* along the signal trajectories, corresponding to the noise-levels, we see that the high noise-levels *did not de-couple* the dynamics between the Driver and the Receiver – it simply added variance at each time-step.

Table 4: Signal Recovery in Driver-Receiver System.

Source	Target	MSE	Variance
Signal	Driver Output	28.0	189
Signal	**Receiver Msg.**	$\mathbf{2e-4}$	$\mathbf{1.1e-10}$
Driver Output	Receiver Msg.	27.1	186

7. Conclusion

In this work, we introduced a novel cryptographic framework for protecting ECU networks from an especially insidious class of attacks: nudging attackers. The proposed framework embeds ECU signals in a VANET into a synchronized set of chaos dynamics. We introduced a novel mechanism for dynamically salting the encryption blocks, making the chaos dynamics computationally unpredictable using state-of-the-art approximation methods. We further leverage the dynamical systems framework to introduce dynamic synchronization of the ECU network connectivities in order to maximize communication efficiency while minimizing excess CAN Bus exposure to ECU communication channels. Our experiments demonstrate a proof-of-concept for the proposed encryption blocks, illustrating their capacity to mask periodic signals. We also show adaptability of the synchronization mechanism on various temporal signal types.



(a) Driver Signal (b) Receiver Recovery

(c) Driver Signal (d) Receiver Recovery

(e) Driver Signal (f) Receiver Recovery

Some notable limitations exist with our current approach. First, it is unclear how vehicle-to-vehicle communications will work under this framework, as the vehicles themselves have different identifying public keys (i.e., $x(0), z(0), m(0)$), which are pre-determined by a manufacturer setting. Secondly, cost-benefit analysis for feasibility will need to be done by manufacturers. This work, however, provides a low-cost alternative to current encryption and IDS schemes, which may be better suited for emerging cybersecurity standards [33–35] and regulatory agency guidelines [36]. Finally, future work will explore the generalization of this technique beyond VANETs, where protection of temporal information is necessary.

Acknowledgements

This work has been supported in part by grants from NSF (CNS-1305358 and DBI-0939454), Ford Motor Company, General Motors Research, and ZF; and the research has also been sponsored by Air Force Research Laboratory (AFRL) under agreement numbers FA8750-16-2-0284 and FA8750-19-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL), the U.S. Government, National Science Foundation, Ford, GM, ZF, or other research sponsors.

References

[1] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 2898–2915, 2017.

[2] J. T. Isaac, S. Zeadally, and J. S. Camara, "Security attacks and solutions for vehicular ad hoc networks," *IET communications*, vol. 4, no. 7, pp. 894–903, 2010.

[3] M. Jenkins and S. M. Mahmud, "Security needs for the future intelligent vehicles," tech. rep., SAE Technical Paper, 2006.

[4] D. S. Wilks, "Effects of stochastic parametrizations in the lorenz'96 system," *Quarterly Journal of the Royal Meteorological Society*, vol. 131, no. 606, pp. 389–407, 2005.

[5] S. H. Strogatz and I. Stewart, "Coupled oscillators and biological synchronization," *Scientific American*, vol. 269, no. 6, pp. 102–109, 1993.

[6] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[7] A. K. Jadoon, L. Wang, T. Li, and M. A. Zia, "Lightweight cryptographic techniques for automotive cybersecurity," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[8] B. Jarupan and E. Ekici, "A survey of cross-layer design for vanets," *Ad Hoc Networks*, vol. 9, no. 5, pp. 966–983, 2011.

[9] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, and Y. Tang, "Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks," *Journal of Network and Computer Applications*, vol. 106, pp. 117–123, 2018.

[10] R. Verdult, W. Meng, F. D. Garcia, D. Doozan, B. Ege, W. Enck, A. C. Snoeren, G. Vigna, T. Xie, N. Feamster, *et al.*, "Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer," in *22nd {USENIX} Security Symposium*, pp. 687–702, 2013.

[11] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection.," in *USENIX Security Symposium*, pp. 911–927, 2016.

[12] O. Avatefipour and H. Malik, "State-of-the-art survey on in-vehicle network communication (can-bus) security and vulnerabilities," *preprint arXiv:1802.01725*, 2018.

[13] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, 2004.

[14] S. S. Haykin and S. S. Haykin, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[15] L. M. Pecora and T. L. Carroll, "Master stability functions for synchronized coupled systems," *Physical review letters*, vol. 80, no. 10, p. 2109, 1998.

[16] K. M. Cuomo and A. V. Oppenheim, "Circuit implementation of synchronized chaos with applications to communications," *Physical review letters*, vol. 71, no. 1, p. 65, 1993.

[17] L. Kocarev, "Chaos-based cryptography: a brief overview," *IEEE Circuits and Systems Magazine*, vol. 1, no. 3, pp. 6–21, 2001.

[18] L. Kocarev and S. Lian, *Chaos-based cryptography: theory, algorithms and applications*, vol. 354. Springer Science & Business Media, 2011.

[19] I. P. Cornfeld, S. V. Fomin, and Y. G. Sinai, *Ergodic theory*, vol. 245. Springer Science & Business Media, 2012.

[20] H. Gilbert and H. Handschuh, "Security analysis of sha-256 and sisters," in *International workshop on selected areas in cryptography*, pp. 175–193, Springer, 2003.

[21] S. Ishii and M.-A. Sato, "Reconstruction of chaotic dynamics by on-line em algorithm," *Neural Networks*, vol. 14, no. 9, pp. 1239–1256, 2001.

[22] A. C. Yao, "Theory and application of trapdoor functions," in *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pp. 80–91, IEEE, 1982.

[23] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[24] E. Biham and O. Dunkelman, "A framework for iterative hash functions—haifa," tech. rep., Computer Science Department, Technion, 2007.

[25] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[26] O. Anava, E. Hazan, S. Mannor, and O. Shamir, "Online learning for time series prediction," in *Conference on learning theory*, pp. 172–184, 2013.

[27] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[28] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[29] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions.," *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[31] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[32] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2213, p. 20170844, 2018.

[33] A. Barber, "Status of work in process on ISO/SAE 21434 automotive cybersecurity standard." ISO/SAE. Working Draft, April 2018.

[34] C. Schmittner, G. Griessnig, and Z. Ma, "Status of the development of ISO/SAE 21434," in *Proc. 25th European Conference, EuroSPI*, pp. 504–513, 01 2018.

[35] Vehicle Cybersecurity Systems Engineering Committee, "J3061 cybersecurity guidebook for cyber-physical vehicle systems," tech. rep., SAE International, 2016.

[36] J. C. Suchodolski, "Cybersecurity of autonomous systems in the transportation sector: An examination of regulatory and private law approaches with recommendations for needed reforms," *NCJL & Tech.*, vol. 20, p. 121, 2018.